Developer Introduction to Workflows for Windows SharePoint Services 3.0 and SharePoint Server 2007

**Summary**: Get a high-level overview of how Microsoft Windows SharePoint Services 3.0 implements the workflow capabilities of the Windows Workflow Foundation, and how Microsoft Office SharePoint Server 2007 extends those capabilities with symmetrical Microsoft Office InfoPath 2007 forms. (28 printed pages)

Andrew May, Microsoft Corporation

May 2006

**Applies to:** Microsoft Windows SharePoint Services 3.0, Microsoft Office SharePoint Server 2007

**Contents**

* [Introduction to Workflows](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__introduction)
* [Workflow Architecture](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__workflowarch)
* [Workflow Types](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__workflowtypes)
* [Workflow Composition](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__workflowcomposition)
* [Workflow Markup](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__workflowmarkup)
* [Workflows in Windows SharePoint Services and SharePoint Server 2007](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__workflowsinwinsharepointandsharepointserver)
* [Authoring SharePoint Workflows](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__authsharepointworkflows)
* [Authoring SharePoint Workflows in Visual Studio 2005](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__authsharepointworkflowsinvs2005)
* [Authoring SharePoint Workflows in SharePoint Designer 2007](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__authsharepointworkflowsinsharepointdes2007)
* [Comparing the Visual Studio 2005 Designer for Windows Workflow Foundation and SharePoint Designer 2007](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__compvs2005designerforwinwfandsharepointdesigner)
* [Workflow Structure in Windows SharePoint Services](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__workflowstrucinwinsharepointservices)
* [Using the Workflow Namespace](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__usingworkflownamespace)
* [Conclusion](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__conclusion)
* [Additional Resources](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__additionalresources)

**Introduction to Workflows**

Microsoft Windows SharePoint Services provides a robust, customizable work environment for users to create, collaborate, and store valuable business information. Now, with Microsoft Windows SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007, you can attach custom business processes to these documents or list items.

You can represent these custom business processes by using workflows. A workflow is a natural way to organize and run a set of work units, or activities, to form an executable representation of a work process. This process can control almost any aspect of an item in Windows SharePoint Services, including the life cycle of that item. The workflow is flexible enough to model both the system functions and the human actions necessary for the workflow to complete.

You can create workflows that are as simple or complex as your business processes require. You can create workflows that the user initiates, or workflows that Windows SharePoint Services automatically initiates based on some event, such as when an item is created or changed.

Suppose you need to create a simple workflow that routes a document to a series of users for approval or comments. This workflow would include actions that the system needs to perform, as well as provide interfaces for the users to interact with the workflow in prescribed ways. For example, Windows SharePoint Services would send an e-mail message to the selected users when the document was ready for review. Those users would then need to be able to notify Windows SharePoint Services when they had completed their reviews and, optionally, enter any comments. The workflow framework included in Windows SharePoint Services 3.0, and extended in SharePoint Server 2007, enables you to model such complex work processes and present them to end users in an easily understood, unobtrusive manner that guides them through each step of the process.

This article provides a high-level overview of workflows as they are implemented in Windows SharePoint Services and extended in SharePoint Server. It includes a discussion of the developer tools available to create workflows in both environments, and the respective capabilities and advantages of those tools.

**Workflow Architecture**

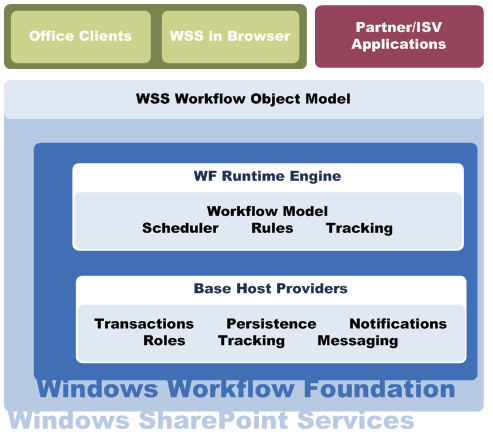
The workflow functionality in Windows SharePoint Services 3.0 is built on the Windows Workflow Foundation (WF), a Microsoft Windows platform component that provides a programming infrastructure and tools for development and execution of workflow-based applications. WF simplifies the process of asynchronous programming to create stateful, long-running, and persistent workflow applications. The WF run-time engine manages workflow execution and allows workflows to remain active for long periods of time and to survive restarting the computer. Run-time services offer functionality such as transactions and persistence to manage errors gracefully and correctly.

The WF run-time engine provides the services that every workflow application needs, such as sequencing, state management, tracking capabilities, and transaction support. The WF run-time engine serves as a state machine responsible for loading and unloading workflows, as well as managing the current state of any workflows that are running. WF allows any application process or service container to run workflows by hosting WF — that is, loading WF within its process.

Windows SharePoint Services hosts the WF run-time engine. In place of the pluggable services that are included with WF, Windows SharePoint Services provides custom implementations of the following services for the engine: transaction, persistence, notifications, roles, tracking, and messaging. Developers can then create workflow solutions that run within Windows SharePoint Services.

Figure 1 shows the workflow architecture in Windows SharePoint Services. Windows SharePoint Services hosts the WF run-time engine within its process, and provides custom implementations of the necessary services. The functionality of the WF run-time engine, as well as the hosting functionality Windows SharePoint Services provides, is exposed through the Windows SharePoint Services object model.

**Figure 1. Workflow architecture in Windows SharePoint Services 3.0**



Windows Workflow Foundation also provides the Visual Studio 2005 Designer for Windows Workflow Foundation, an add-in hosted within the Microsoft Visual Studio 2005 development system that enables developers to create their own custom workflows and workflow activities. For more information, see [Authoring SharePoint Workflows in Visual Studio 2005](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__authsharepointworkflowsinvs2005).

SharePoint Server employs the workflow functionality in Windows SharePoint Services 3.0, and extends that functionality through integration with InfoPath forms, and additional workflow activities. This is discussed in greater detail in [Authoring SharePoint Workflows in SharePoint Designer 2007](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__authsharepointworkflowsinsharepointdes2007).

The Windows Workflow Foundation run-time engine is available as part of the [Microsoft Windows Workflow Foundation Runtime Components Beta 2.2 and Visual Studio 2005 Extensions for Windows Workflow Foundation Beta 2.2](http://www.microsoft.com/downloads/details.aspx?familyid=5c080096-f3a0-4ce4-8830-1489d0215877&displaylang=en) download available on the Microsoft Download Center. This download also contains the Visual Studio 2005 Designer for Windows Workflow Foundation, as well as the Windows Workflow Foundation Software Development Kit (SDK).

Now that we have briefly described the workflow infrastructure provided in Windows SharePoint Services, let's examine the internal composition and structure of a WF workflow itself. After that, we'll take a look at the various developer tools Microsoft provides for creating workflow solutions for Windows SharePoint Services and SharePoint Server 2007.

**Workflow Persistence**

One of the most important services Windows SharePoint Services provides to the WF workflow engine is that of *persistence*. Workflows that include human interaction are inherently long running; even in ideal circumstances, humans require a relatively long time to complete work compared to machines. In many Microsoft Office scenarios, workflows typically require days, if not longer. Consider the example workflow that routes documents for approval. The approver might need several days to review the document.

Clearly, leaving each running workflow in memory for the entire duration of its execution is not feasible; very soon the resources required by accumulated long-running workflows would bring the system to a halt.

Instead, after a workflow instance has reached a point at which it is waiting for user input, Windows SharePoint Services unloads that workflow instance from memory and persists its data. Then, when an appropriate event, such as the user entering input, requires that the workflow instance start again, Windows SharePoint Services re-creates the workflow instance using the persisted data, so the workflow instance can receive and handle the event as necessary.

Although numerous workflow instances may run at any given time, only a fraction of those workflows are likely to be in memory and use system resources.

**Workflow Types**

Windows Workflow Foundation supports two fundamental workflow styles:

* **Sequential workflows**   Represents a workflow as a procession of steps that execute in order until the last activity completes. However, sequential workflows are not purely sequential in their execution. Because they can receive external events and include parallel logic flows, the exact order of activity execution can vary.
* **State machine workflows**   Represents a set of states, transitions, and actions. One state is denoted as the start state, and then, based on an event, a transition can be made to another state. The state machine can have a final state that determines the end of the workflow.

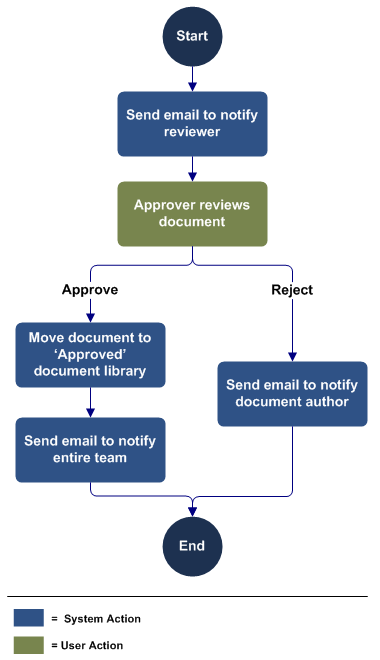
You can create workflows of either type for Windows SharePoint Services and SharePoint Server.

**Sequential Workflows**

Sequential workflows can best be represented graphically as a flowchart of actions, with a beginning, an end, and a sequential flow direction from start to finish. Sequential workflows can incorporate flow structures such as repetition, looping, and parallel branches, but ultimately progress from the initial action to the final action.

For example, suppose you were to chart the simple workflow that routes a document in Windows SharePoint Services for approval. When the workflow starts, the system notifies the specified reviewer, by e-mail message, that he or she has a document to review. The reviewer then reviews the document and notifies the system that the task is completed and whether the reviewer approves or rejects the document. Based on the reviewer response, the workflow executes one of two parallel branches. If the reviewer approved the document, the system moves the approved document to a specific SharePoint document library and then sends an e-mail message to the entire team notifying them of the approved document. If the reviewer rejects the document, the system notifies the document author. In either case, the workflow then reaches its end and terminates. Figure 2 shows this workflow.

**Figure 2. Conceptual diagram of a sequential workflow**



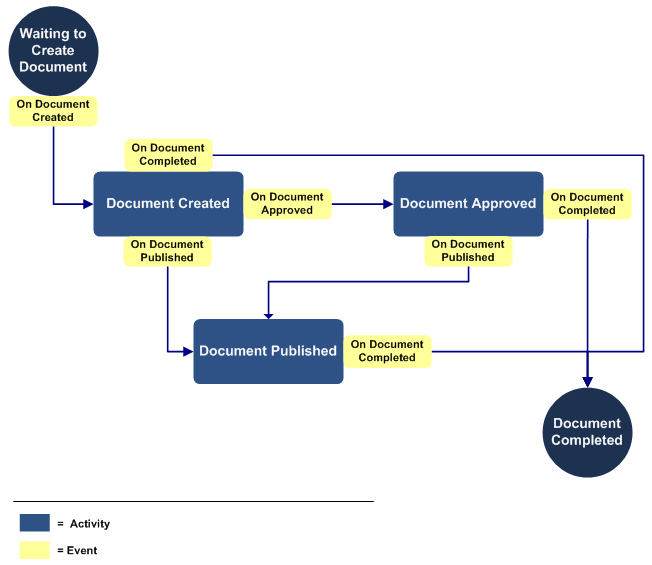
**State Machine Workflows**

Unlike sequential workflows, state machine workflows do not have a prescribed execution flow, and need not have an end. Instead, state machine workflows define any number of states which an item may inhabit, and the events that transition the item from one state to another.

Figure 3 represents a simple document publishing process, modeled as a state machine workflow. The workflow is initiated when a document is created, and ends when the document state is set as completed. In between, however, the document does not travel in a predetermined path, but instead transitions from state to state as events occur.

The state machine workflow is composed of *state activities*. Each state activity represents a state for the item. Each state activity can contain optional state initialization, state finalization, and one or more event handlers. Each event handler activity can handle one event. In response to the event handled, some processing can be done, and a transition can be made to another state.

**Figure 3. Conceptual diagram of a state machine workflow**



One characteristic of both sequential and state machine workflows is that each type of workflow can be broken into a collection of discrete actions—those performed by the system, and those taken by users. You can consider these actions to be the building blocks of workflows. In WF workflows, these actions are referred to as *activities*.

**Workflow Composition**

Earlier, we defined a workflow as a way to organize and run a set of work units, or activities, to form an executable representation of a work process. Therefore, each WF workflow consists of a set of related activities. An activity is the elemental unit of modeling, programmability, reuse, and execution within Windows Workflow Foundation. An activity may be performed by the system or by a user. For example, the system may send someone an e-mail message as an alert; or a person might approve a document for distribution.

Conceptually, our sequential workflow example includes five activities:

* Send e-mail message to notify approver
* Approve or reject document
* Move document to "Approved" document library
* Send e-mail message to notify team
* Send e-mail message to notify document author

Activities can also represent logical control structures that define scope and direct the execution flow of the workflow, much as code logic controls, such as **If Then** and **Do While** loops, control the program flow in code.

Activities can have properties, methods, and events. *Simple activities* perform a single unit of work, such as "delay for 1 day" or "invoke Web service." *Composite activities* contain other activities; for example, a conditional with two branches. You can also attach handlers, such as error or compensation handlers, to activities. This is especially advisable for composite activities.

In essence, the workflow itself is simply a composite activity that contains all the other activities in that workflow.

Therefore, workflow can be seen as a set of activities stored as a model that describes a real-world process. Work passes through the model from start to finish, and activities might be executed by people or by system functions. Workflow provides a way of describing the order of execution and dependent relationships between pieces of short-running or long-running work.

Windows Workflow Foundation includes a number of predefined activities that you can use in your workflows, or you can create your own custom activities. The Windows SharePoint Services workflow project template pack contains numerous activities designed explicitly for use in the Windows SharePoint Services environment. The template pack also contains the references necessary to program against the Windows SharePoint Services object model. Similarly, the SharePoint Server project template pack is customized for use in the SharePoint Server environment. For more information, see [Authoring SharePoint Workflows in Visual Studio 2005](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__authsharepointworkflowsinvs2005).

**Workflow Markup**

Each WF workflow can be represented by the following combinations of files:

* An XML file, or markup file, that includes the declarative metadata of the workflow
* The markup file, in combination with a code-behind file that contains custom code representing the properties and behavior of the workflow
* A code file (or files) that include both the declarative logic and behavior of the workflow

The markup file is written in Extensible Application Markup Language (XAML), which has a published schema to which the file must adhere, and given a file extension of .xoml.

Because the XAML has a published schema, you can create XAML files using any text or XML editor you choose. However, the two workflow authoring tools discussed in this article, the Visual Studio 2005 Designer for Windows Workflow Foundation and Microsoft Office SharePoint Designer 2007, provide developers with a graphical interface in which to create workflows and automatically generate the appropriate markup file.

Developers can choose to integrate or separate their declarative metadata from the business logic included in the workflow. Conceptually, the "code separation" paradigm that WF workflows employ is similar to that used in Microsoft ASP.NET: declarative metadata is separate from the file that encapsulates your business logic. So although the markup file contains the metadata for the activities included in the workflow, the properties and behaviors of those activities are detailed in a separate file.

For workflows authored using code separation, information is persisted in the markup file, as detailed previously, and in one of the following two types of files:

* A code-beside file, which contains the code that encapsulates your business logic. This file may be written in either Microsoft Visual C# or Microsoft Visual Basic .NET.
* A workflow rules file, which encapsulates your business logic in declarative rules, rather than code.

Each workflow created this way is actually a unique Microsoft .NET type, constructed from two partial classes, which are represented by the XOML and a code-behind or rules file. When the workflow project is compiled, these two partial classes are combined into a .NET assembly. This is the approach you take when authoring workflows for Windows SharePoint Services or SharePoint Server by using the Visual Studio 2005 Designer for Windows Workflow Foundation.

Workflows composed only of code files follow the same general compilation process — that is, the code files are compiled into a .NET type.

In addition, you can compile workflows composed solely of markup files. However, this is not necessary; the WF run-time engine can load and run uncompiled markup workflows. This is the approach you take when authoring workflows for Windows SharePoint Services or SharePoint Server by using SharePoint Designer 2007.

**Workflows in Windows SharePoint Services and SharePoint Server 2007**

So far, we have discussed the attributes of workflows for any application that implements the Windows Workflow Foundation. Now, let's take a closer look at workflows as they are implemented in Windows SharePoint Services specifically and discuss important aspects of that implementation.

**Workflow Templates and Instances**

In Windows SharePoint Services, workflows that are available on a site or list are referred to as workflow *templates*; a workflow that is actually running on a specific SharePoint item is called a workflow *instance*. So, on a given list, several instances of the same workflow template may be running at the same time, each on a different SharePoint item. More than one workflow can be running on a given SharePoint item at a time.

You can run workflows on documents or list items. You make a workflow template available on a site through a process called *association*, discussed later in this article.

**Using Forms to Enable User Interaction with Windows SharePoint Services**

Although you can use Windows SharePoint Services workflows to model any number of unique business processes, there are several common stages at which user can interact with the workflow template or instance.

For the user to pass information to the workflow, the developer must provide an interface for that interaction through the use of a form. Adding forms to workflows enables you to make your workflows more dynamic and flexible. Forms enable you to gather information from users at predefined times in the life of a workflow, as well as let users interact with the tasks for that workflow.

Technically, you can employ any forms technology with WF workflows, as long as your forms are capable of doing the following:

* Invoking the Windows SharePoint Services object model
* Generating the data necessary to send to the Windows SharePoint Services object model
* Receiving and parsing the required data from the Windows SharePoint Services object model

Any information passed to the form on load is formatted as a string, as is the data the form must pass back to the Windows SharePoint Services 3.0 object model when the user submits the form. Although this string is typically XML, you can use any data format that can be formatted as a string, as long as your form is able to generate strings in that format and parse strings it receives.

Windows SharePoint Services supports the use of ASP.NET 2.0 forms. How these forms are implemented depends on whether the developer uses the Visual Studio 2005 Designer for Windows Workflow Foundation or SharePoint Designer 2007 as the workflow authoring tool. In addition, SharePoint Server 2007 offers developers the ability to integrate Microsoft Office InfoPath forms with their workflows. These InfoPath forms can be hosted within the Microsoft Office client applications, such as Microsoft Office Word, Microsoft Office PowerPoint, Microsoft Office Excel, and InfoPath, as well as the client browser, and provide a richer client experience. For more information about how each workflow authoring tool implements workflow forms, see [ASP.NET Forms in Windows SharePoint Services Workflows](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__aspnetformsinworkflows), [InfoPath Forms in SharePoint Server Workflows](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__infopathformsinworkflows), and [ASP.NET Forms in SharePoint Designer Workflows](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__aspnetformsinsp).

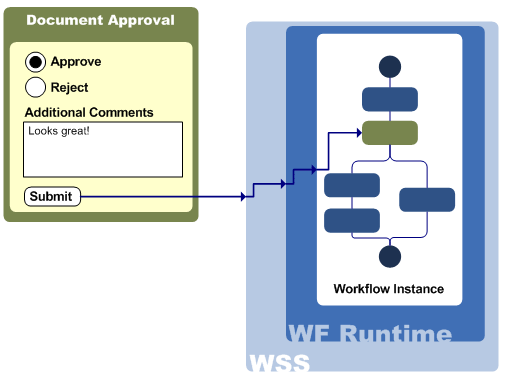
Whatever forms technology is used, the conceptual operation of each form is the same. Windows SharePoint Services displays the form to the user at the appropriate juncture in the workflow. The user enters information, and submits the form.

As workflow developer, you are responsible for programming what happens when the user submits a form. In most cases, your form calls into the Windows SharePoint Services object model and invokes the appropriate method to pass the form information to the correct workflow instance.

The information string can be in any format, provided that both the form and the workflow activity are programmed to parse the string. In most cases, developers will likely choose to use XML.

Windows SharePoint Services passes the string information to the WF run-time engine, the workflow instance, and ultimately the correct activity within the workflow instance. The activity receiving the information can then respond as programmed. Figure 4 shows how a form integrates with workflow.

Figure 4. Form integration with workflow



Three types of forms are used in Windows SharePoint Services workflows:

* **Association and initialization forms**   Association and initialization forms are displayed for users to fill out before any workflow is actually started. You can use these forms to enable users to set parameters and other information for the workflow before it starts.
* **Modification forms**   Modifications are options you present to users to change the workflow as it is running on an item. You can then create modification forms that enable users to specify the parameters of the modification.
* **Task forms**   You can also specify custom forms for the tasks in your workflow, based on the workflow task type. Each workflow task type in Windows SharePoint Services is assigned a content type; the content type definition determines whether a custom form is specified for a given workflow task type.

These three types of forms represent the predefined stages at which users can interact with the workflow. We discuss these stages next.

**Points of User Interaction with Workflows**

Let's look more closely at the various stages at which users can interact with workflows in Windows SharePoint Services and SharePoint Server.

**Association**

Association occurs when a site administrator makes the workflow template available on a specific document library, list, or content type. At this stage, the site administrator customizes the workflow for this specific document library, list, or content type by specifying the following parameter information:

* A unique name for the workflow
* How the workflow is applied to a given item: either automatically when the item is created or changed, or manually; and which roles, such as Administrator or Contributor, can start a workflow
* The Task list, in which the workflow can create tasks
* The History list in which the workflow can store history events, as defined by the workflow

In addition, the workflow developer can enable the site administration to further customize the workflow by setting parameter information specific to the particular workflow. The administrator may have to specify parameters, default values, and other information for the workflow as it applies to items on the list, library, or content type with which the administrator is associating it.

Associating a workflow actually occurs outside the workflow itself; no workflow instance is actually started during association. Rather, Windows SharePoint Services stores the association information in a special internal workflow association table. Then, when the workflow instance is actually started, Windows SharePoint Services uses the association data (and the initiation data, if any) to set the parameters of the new workflow instance.

**Initiation**

Whereas association deals with the workflow as it applies to a specific list, library, or content type, initiation deals with the workflow as it applies to a specific SharePoint item.

Initiation occurs when a user actually starts a workflow instance on a specific item. The user supplies information custom to that specific workflow on that item, if necessary, and then starts the workflow.

Workflow developers can use initiation to let users override or append the association parameters set by administrators, or specify additional parameters or information about the workflow as it applies to the given SharePoint item. Not all workflows need initiation. In fact, a workflow set to start automatically cannot have initiation parameters.

The Windows SharePoint Services workflow project for the Visual Studio 2005 Designer for Windows Workflow Foundation includes an activity that acts as the event handler when the workflow is initiated. This activity is the first activity in any Windows SharePoint Services workflow.

**Modification**

Modifications enable users to alter the parameters or flow of a workflow instance as it is running. As a developer, you may have points in your workflow at which you would like to enable users to alter the workflow while it is running on an item. For example, you might want to enable a user to assign a task to another person, or even add a specific task to the workflow. The options you present to users to change the workflow as it is running on an item are called *modifications*.

The Windows SharePoint Services workflow project for the Visual Studio 2005 Designer for Windows Workflow Foundation includes an activity to enable a workflow modification, as well as an activity that acts as an event handler when a workflow modification is enabled.

**Tasks**

Workflow activities that are performed by people are represented as tasks in Windows SharePoint Services workflows.

As workflow author, you can specify the task schema. For example, the task listing could include the following:

* Task title
* Name of person to whom the task is assigned
* Task status
* Task priority
* Task date due
* A link to the referenced item

The user can then edit the task as necessary. The workflow instance is notified of changes to workflow tasks and can choose to respond to them as specified in the workflow.

The Windows SharePoint Services workflow project for the Visual Studio 2005 Designer for Windows Workflow Foundation includes activities that create, delete, and update tasks, as well as activities that act as event handlers when tasks are created, updated, or deleted.

**Authoring SharePoint Workflows**

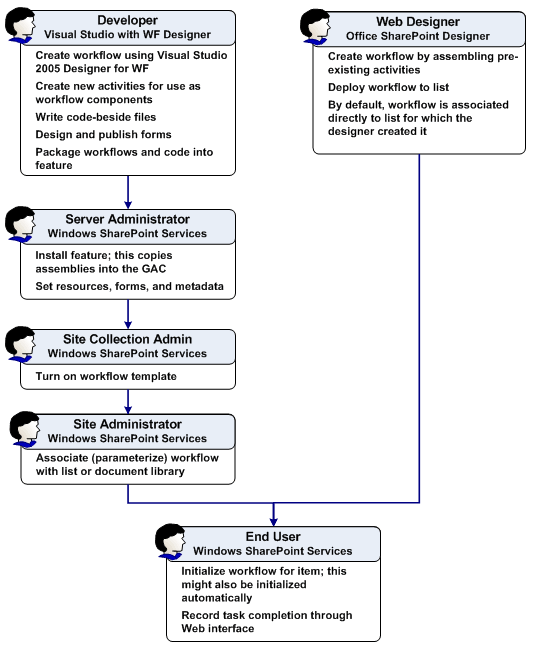
Microsoft provides two development tools for authoring workflows for Windows SharePoint Services: the Visual Studio 2005 Designer for Windows Workflow Foundation, and Office SharePoint Designer 2007. Because each authoring tool produces workflows with different attributes and capabilities, it is worth examining each tool in detail.

Figure 5 shows the various steps that need to be performed to create, deploy, associate, and run a workflow with each of the authoring tools. In general, the largest differences between the two tools are the following:

* Workflow authoring in the Visual Studio 2005 Designer for Windows Workflow Foundation is performed by a professional developer, who creates a workflow template that can be deployed across multiple sites and contains custom code and activities. The developer then passes the workflow template to a server administrator for actual deployment and association.
* Workflow authoring in SharePoint Designer is likely done by someone other than a professional developer, such as a Web designer or knowledge worker, who wants to create a workflow for a specific list or document library. In this case, the designer is limited to the workflow activities on the Safe Controls List, and the workflow cannot include custom code. The workflow author deploys the workflow template directly to the list or document library as part of the workflow authoring process.

For a detailed comparison of the capabilities of the two tools, see [Comparing the Visual Studio 2005 Designer for Windows Workflow Foundation and SharePoint Designer 2007](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__compvs2005designerforwinwfandsharepointdesigner).

**Figure 5. Workflow authoring, deployment, and initiation process**



**Authoring SharePoint Workflows in Visual Studio 2005**

The Visual Studio 2005 Designer for Windows Workflow Foundation is an add-in for Visual Studio 2005. It provides a means to rapidly develop workflows by using a graphical interface that leverages a developer's knowledge of the Visual Studio development environment.

Visual Studio 2005 Designer for Windows Workflow Foundation is a tool for you to rapidly create the workflow in a way that is integrated with the development of the code encapsulating your business processes. To do this, the Visual Studio 2005 Designer for Windows Workflow Foundation provides a graphical interface with intuitive controls, hosted within the familiar Visual Studio development environment. Its features include the following:

* A drag-and-drop design surface that lets you assemble custom workflows from predefined workflow activities you drag from the **Toolbox**
* An interface that lets you work on your workflow markup using intuitive graphical tools
* Integration with the **Properties** window, so developers can configure properties of workflow activities either through the graphical interface or directly in the code-beside file, and always keeps the two in sync
* Debugging of your workflows by attaching to the Windows SharePoint Services process, including setting breakpoints in your workflow
* The ability to attach error, compensation, and event handlers to activities, and "comment out" activities graphically

The Visual Studio 2005 Designer for Windows Workflow Foundation is available as part of the [Microsoft Windows Workflow Foundation Runtime Components Beta 2.2 and Visual Studio 2005 Extensions for Windows Workflow Foundation Beta 2.2](http://www.microsoft.com/downloads/details.aspx?familyid=5c080096-f3a0-4ce4-8830-1489d0215877&displaylang=en) download on the Microsoft Download Center. This download also contains the Windows Workflow Foundation run-time engine, as well as the Windows Workflow Foundation SDK.

To further assist your workflow development, Microsoft provides two Visual Studio project template packs for use with the Visual Studio 2005 Designer for Windows Workflow Foundation: one customized for Windows SharePoint Services workflows, the other customized for creating workflows for SharePoint Server.

The Window SharePoint Services workflow project template contains a reference to the Windows SharePoint namespace and includes custom workflow activities designed specifically for the Windows SharePoint Services environment. These custom activities enable you to perform functions common to the Windows SharePoint Services environment, such as creating, updating, completing, and deleting SharePoint tasks, sending e-mail, and enabling workflow modifications.

The SharePoint Server workflow project contains everything the Window SharePoint Services workflow project template, as well as a reference to the SharePoint Server namespace, and additional functionality for managing workflow tasks in the SharePoint Server work environment.

**The Workflow Development Process in Visual Studio**

In general, when you develop workflows for Windows SharePoint Services or Office SharePoint Server using the Visual Studio 2005 Designer for Windows Workflow Foundation, you follow these basic steps:

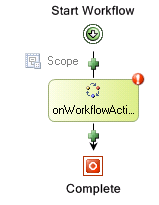
1. Author your workflow, including the code-beside file if needed, in the Visual Studio 2005 Designer for Windows Workflow Foundation.
2. Design and publish any forms you want to use with your workflow.
3. Author the feature definition and workflow template definition file, which contains information about the workflow assembly, and binds the forms to the workflow assembly.
4. Compile the workflow files into a .NET assembly.
5. Package the workflow assembly and workflow definition together and deploy them by using the features functionality in Windows SharePoint Services.
6. Debug the live workflow assembly using the Visual Studio 2005 Designer for Windows Workflow Foundation.
7. Recompile and deploy the workflow assembly as necessary to fix any bugs found.

The following sections provide an overview of each of these development steps.

**Creating Workflows by Using the Visual Studio 2005 Designer for Windows Workflow Foundation**

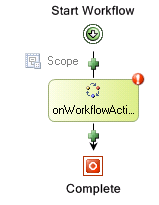
When you select a new Windows SharePoint Services or Office SharePoint Server workflow project, Visual Studio displays the Visual Studio 2005 Designer for Windows Workflow Foundation design surface (Figure 6). This design surface presents a graphical interface, where you can assemble workflows by dragging and dropping various activities from the **Toolbox**.

**Figure 6. Visual Studio 2005 Designer for Windows Workflow Foundation interface**



When you drag a specific activity onto your workflow, the Visual Studio 2005 Designer for Windows Workflow Foundation shows you the valid locations for that activity within the workflow. You cannot place an activity in a location where it would not be valid. For example, you cannot position a Send activity as the first activity in a Listen activity branch. As shown in Figure 7, Visual Studio 2005 Designer for Windows Workflow Foundation displays green plus-sign icons to denote valid positions for the specific activity.

**Figure 7. Valid positions for a Workflow activity**



As you graphically design your workflow, the Visual Studio 2005 Designer for Windows Workflow Foundation actually generates corresponding markup.

In addition, if you are working with code separation, your workflow project contains a code-behind file, in which you program the business logic of your workflow. You can switch between working in the workflow design surface and the code-beside file whenever you want.

**Setting Workflow Properties**

After you add the activity to the workflow, you must set the properties of the activity for it to be valid within the workflow. You can set these properties by using the standard Visual Studio Properties window. The type of data you can specify depends on the type of data the property itself accepts: literal values, variables, or method names.

If you want to specify a variable for the property, you can either enter the variable name in the Properties window, in which case the Visual Studio 2005 Designer for Windows Workflow Foundation automatically declares the variable in the code-behind file; or declare the variable in the code-behind file, and then select it in the Properties window.

Certain activity properties are essentially references to methods in the code-beside file that conform to a specific signature. As with variable names, you can enter the method name in the Properties window and let the Visual Studio 2005 Designer for Windows Workflow Foundation add the method signature to the code-behind file, or you can create the method in the code-behind file and then select the method in the Properties window.

You can also bind a property to the property of another activity.

**Handler Activities**

A workflow might have several potential points of failure. It is important to track the state of a workflow and to report errors as they occur, so that you can resolve problems accurately and with a minimum of effort. It is equally important for a workflow to maintain the integrity of a set of closely related actions, so that if part of an operation takes place but another does not, the entire operation can be rolled back. You can use the **FaultHandlerActivity**, **TransactionScopeActivity**, **CompensationHandlerActivity**, and **CancellationHandlersActivity** activities to handle errors, to maintain the state of a workflow, and to fix problems as they occur.

You can think of a **FaultHandlersActivity** activity as a C-language **try** block, to which you can attach an ordered set of **FaultHandlerActivity** activities that act as exception handlers. These exception handlers can be thought of as C-language **catch** blocks. If an exception is thrown during the execution of a composite activity, the WF run-time engine matches the exception against the exception types that are handled by the **FaultHandlerActivity** activities. If the run-time engine finds no matching exception handler, it then communicates the exception to the next higher composite activity, where the process repeats, and so on, until an appropriate handler is found.

You can also have **EventHandlersActivity** activities that respond to events, by adding event handlers to an **EventHandlerScopeActivity** activity. Conceptually, these event handlers are very similar to those in the C or Visual Basic .NET languages. To create an event handler, you must use **EventDrivenActivity** activities.

**CompensationHandlerActivity** activities contain code that compensates for, or rolls back, the operations of a composite activity if it does not perform successfully.

**ASP.NET Forms in Windows SharePoint Services Workflows**

As mentioned earlier, you can use ASP.NET to create the forms you use with your Windows SharePoint Services workflow. These forms are then displayed in the Windows SharePoint Services user interface at the appropriate stages in the workflow.

Workflow forms are late-bound to the workflow assembly through information you supply in the workflow template definition XML file. The workflow template definition schema contains elements for denoting the URL of the various types of forms you can use with Windows SharePoint Services workflows. This provides the ability to create elements for forms for any custom workflow modifications, as well as forms for the various types of SharePoint tasks used in the workflow.

In most cases, the workflow assembly itself contains no information about or link to the workflow forms themselves. Developers can change the workflow forms to be used just by editing the workflow definition XML, without having to recompile the workflow assembly itself. The one exception is workflow modifications; each activity that enables a workflow modification must contain the GUID of the form for that workflow modification.

**InfoPath Forms in SharePoint Server Workflows**

Although you can also use ASP.NET workflow forms in SharePoint Server workflows, SharePoint Server offers you the capability to extend your workflow forms and display them in Microsoft Office client applications.

You can use InfoPath forms with your workflows. InfoPath gives you the ability to create symmetrical forms — that is, forms that look and operate exactly the same whether they are displayed in the SharePoint Server Web interface or within a Microsoft Office client application, such as Word, Excel, or PowerPoint. This provides a richer interaction experience in which the user can interact with the workflow directly in the client application, rather than having to leave the client and switch to the SharePoint Server Web interface. And as a developer, you don't have to create two separate forms, one for use on the server and one for use in the client, to provide the user with this client application integration.

For more information about creating symmetrical forms in general, refer to the Microsoft Office InfoPath 2007 client help or [Office Online](http://office.microsoft.com/en-us/fx010857921033.aspx).

SharePoint Server uses Office Forms Services, a server-based run-time environment for InfoPath forms, to host workflow forms. Office Forms Services consumes the forms you create in the InfoPath client application and renders them in an ASP.NET framework, which acts as a run-time environment for the form. This environment presents a form editing experience that matches the InfoPath client application.

For more information about Office Forms Services, see the [Microsoft Office SharePoint Server 2007 SDK](http://msdn2.microsoft.com/en-us/library/ms550992.aspx).

|  |
| --- |
| **NoteNote:** |
| Because Office 2007 client applications such as Word, PowerPoint, and Excel include functionality for hosting InfoPath forms, the user does not need to have the InfoPath client application installed to take advantage of this rich integration. |

**Displaying InfoPath Workflow Forms**

SharePoint Server uses the same basic technique to display all custom InfoPath workflow forms, including association, initiation, modification, or edit task forms.

When the user clicks a link to display a workflow form in the SharePoint Server interface, SharePoint Server loads an ASPX page that contains an Office Forms Services Web Part. This Web Part then converts the appropriate InfoPath form into ASP.NET and loads it. When the user submits that form, the Web Part receives the data from the form and handles it accordingly.

The ASPX pages that contain the Office Forms Services Web Part are included in SharePoint Server.

**Specifying InfoPath Workflow Forms**

Again, you specify the custom forms you want to use in the workflow template definition, rather than the workflow itself. In most cases, this involves setting two elements. First, you set the form URL for that workflow process (association, initiation, modification, and so forth) to the appropriate ASPX hosting page included with SharePoint Server. Next, you add an element specifying the URN for the custom InfoPath form for that type of workflow process.

**Submitting Information by Using InfoPath Workflow Forms**

For the hosting ASPX page to receive data from the hosted form, the developer adds a **Submit** button to the InfoPath form. This button uses a rule to submit data by using the data connection to the hosting environment. This connection passes the data back to the hosting ASPX page automatically when the user clicks the **Submit** button. The hosting ASPX page then handles parsing the data and passing it back to the workflow or document library, as appropriate.

**Deploying Workflows**

When you are finished specifying your workflow, you can choose to compile your workflow as a workflow, or as an activity.

After you have compiled your workflow, you can use the SharePoint Features functionality to package and deploy the workflow assembly and any necessary supporting files.

Feature packaging is a way of encapsulating Windows SharePoint Services solutions and functionality for ease of deployment. It provides a mechanism by which developers can package the files needed for a solution, such as workflows, Web Parts, lists, and site definitions, for ease of distribution and deployment. Developers package the necessary files into a .wsp file, which is essentially a .cab file containing a manifest that lists its contents. For more information about SharePoint Features, see the Microsoft Windows SharePoint Services 3.0 SDK.

Each workflow template you deploy must contain a workflow template definition file. A workflow template definition is an XML file that contains the information that Windows SharePoint Services requires to instantiate and run the workflow, such as the following:

* The name, GUID, and description of the workflow template
* Assembly
* The URLs (or URNs for IP forms) of any custom forms used with this workflow template
* Optionally, the names of the workflow, workflow engine, and host service assemblies to use when running the workflow
* Optionally, the correct classes within those assemblies to call

The workflow assembly itself must be deployed to the global assembly cache.

After it is deployed to a site, the workflow is available as a workflow template, which SharePoint administrators can associate with the document libraries and lists on that site.

**Debugging Workflows**

After you deploy your workflow assembly, you can debug your workflow by opening your workflow project and attaching to the Windows SharePoint Service *w3wp* process.

Because the Visual Studio 2005 Designer for Windows Workflow Foundation is hosted within Visual Studio, you can take full advantage of the debugging capabilities of Visual Studio. You can set breakpoints both in the code you write in your code-beside file, as well as on workflow activities in the design surface.

|  |
| --- |
| **NoteNote:** |
| For ease of debugging, we strongly recommend that you develop your workflow templates on a server to which you plan to deploy them. |

The Visual Studio 2005 Designer for Windows Workflow Foundation not only supports standard Visual Studio debugging features such as the Breakpoints and Call Stack windows, it also includes a range of visual indicators that provide information during the debugging process. You can also add breakpoints to a workflow activity while you are debugging the assembly.

You can perform step in, step out, and step over operations to move through workflow code.

The following types of debugging are not supported by the Visual Studio 2005 Designer for Windows Workflow Foundation:

* Just-in-time debugging of run-time exceptions in the hosting process
* Just-in-time debugging by selecting a process in the Task Manager

For more information about debugging using the Visual Studio 2005 Designer for Windows Workflow Foundation, see the Windows Workflow Foundation SDK.

|  |
| --- |
| **NoteNote:** |
| The Windows Workflow Foundation SDK is available as part of the [Microsoft Windows Workflow Foundation Runtime Components Beta 2.2 and Visual Studio 2005 Extensions for Windows Workflow Foundation Beta 2.2](http://www.microsoft.com/downloads/details.aspx?familyid=5c080096-f3a0-4ce4-8830-1489d0215877&displaylang=en) download available on the Microsoft Download Center. This download also contains the Visual Studio 2005 Designer for Windows Workflow Foundation, as well as the Windows Workflow Foundation run-time engine. |

**Authoring SharePoint Workflows in SharePoint Designer 2007**

When you author a workflow in Office SharePoint Designer 2007, you are authoring that workflow directly against, and data-binding to, a specific list or document library in Windows SharePoint Services. You are using a predefined list of workflow activities and are not using any code. The workflow you design is not compiled as an assembly; it is stored as source files until Windows SharePoint Services compiles it the first time it runs.

Such an approach has several advantages:

* Workflows can be developed and tested rapidly.
* Because the workflow is specific to a given list, the deployment process is much less complicated.
* Security issues are much less complicated as well, for the same reason.
* Because they are not compiled into assemblies, workflows created in SharePoint Designer can be deployed to servers where administrative policy prohibits custom code assemblies.

|  |
| --- |
| **NoteNote:** |
| Workflows authored in SharePoint Designer are assembled from a "safe list" of predefined activities, which presumably have been approved by administrators to run on the servers. |

* Workflows can be created by users with less developer experience, such as Web designers or knowledge workers.

This approach also means that workflows authored in SharePoint Designer differ from those created using the Visual Studio 2005 Designer for Windows Workflow Foundation in several important ways:

* A workflow authored in SharePoint Designer cannot be deployed to multiple lists. It is valid only for the list for which it was created.
* Because you author the workflow directly against a list, the workflow is associated with the list at design time. Therefore, workflows authored in SharePoint Designer have no association stage.
* Workflow modifications are not available in workflows authored in SharePoint Designer.
* You cannot author against a content type in SharePoint Designer.

For a detailed comparison, see [Comparing the Visual Studio 2005 Designer for Windows Workflow Foundation and SharePoint Designer 2007](http://msdn2.microsoft.com/en-us/library/aa830816.aspx#office2007ssintrotoworkflows__compvs2005designerforwinwfandsharepointdesigner).

**Running Workflows Authored in SharePoint Designer 2007**

Because they contain no custom code, workflows authored in Office SharePoint Designer are not compiled and deployed as assemblies. Rather, they are stored as their source files within Windows SharePoint Services and compiled into memory only when needed.

For each site, the workflows of this type are stored in a separate document library. This document library contains a folder for each workflow authored in SharePoint Designer. The folder contains all the source files necessary for the workflow, including:

* The workflow markup file
* The workflow rules file
* ASPX forms for any custom workflow forms necessary

Windows SharePoint Services includes a just-in-time compiler to compile the source files into a workflow the first time that workflow is started on an item. Windows SharePoint Services retains the compiled workflow in memory against the time it is called again, similar to how servers keep compiled .aspx pages in memory to speed execution performance the next time a page is called.

Each time a workflow is started on an item, Windows SharePoint Services determines whether the workflow was deployed as an assembly or as source files. If a workflow assembly exists, Windows SharePoint Services calls that assembly to create the workflow instance. If the workflow was deployed as source files, Windows SharePoint Services determines whether it already has a workflow compiled from those source files in memory. If it does, Windows SharePoint Services calls the in-memory compiled workflow to create the workflow instance. If not, Windows SharePoint Services uses its just-in-time compiler to compile the source files into an in-memory workflow, which it then calls to create the workflow instance.

**Workflow Development Process in SharePoint Designer 2007**

To aid the rapid design and deployment of workflows, the development process in SharePoint Designer is much simpler than in Visual Studio.

In general, when you develop workflows for Windows SharePoint Services or SharePoint Server using the Visual Studio 2005 Designer for Windows Workflow Foundation, you follow these basic steps:

1. Author your workflow by assembling and configuring the predefined activities and conditions available in SharePoint Designer.
2. Have SharePoint Designer automatically generate ASP.NET forms for workflow initiation and any custom SharePoint task, if necessary.
3. Customize the workflow forms, if necessary.

SharePoint Designer automatically generates the workflow definition template and handles the deployment of the workflow to the specified list.

The following sections provide an overview of each of these development steps.

**Creating Workflows by Using SharePoint Designer 2007**

SharePoint Designer uses a wizard-driven interface that enables users to assemble sequential workflows from predefined activities. Users select activities from a predetermined list and configure them using the SharePoint Designer interface. These activities can be the same activities present in the Visual Studio 2005 Designer for Windows Workflow Foundation; there is no difference in activities between the two tools.

In SharePoint Designer, however, each activity appears as an action, represented by a sentence that contains variables that the user can configure using drop-down menus and look-up dialog boxes. Users can also select conditions, which are configurable conditional clauses that direct the flow of the workflow.

As the user selects and configures conditions and actions in the workflow interface, SharePoint Designer generates the two files that actually represent the workflow class:

* The workflow markup file, which contains markup that describes the activities included in the workflow
* The workflow rules file, which contains the business logic of the workflow in declarative rules form, rather than code

**Adding Custom Activities and Conditions**

As noted, workflow designers in SharePoint Designer cannot create custom activities for use in their workflows. Instead, they are limited to the activities and conditions that have been made available on the Safe Controls List (presumably approved by a server administrator) that appears in SharePoint Designer.

A condition is simply a custom assembly with a static method that evaluates some condition and returns a Boolean when called.

Developers can create custom activities and conditions, and make them available on the safe list. To do this, developers must:

1. Create the activity or condition, compile it as a strong-named assembly, and deploy it to the global assembly cache.
2. Add the activity or condition to the action safe list in the web.config file.
3. In the WSS.Actions file, located in the workflow folder, add rules and parameters for the sentence that represents the activity or condition in the SharePoint Designer user interface. This is markup that specifies how the activity or condition appears and performs in the interface, because this information is not present in the activity or condition assembly.

**ASP.NET Forms in SharePoint Designer Workflows**

You can create an initiation stage for your workflow in SharePoint Designer. If you do, SharePoint Designer automatically generates an initiation form by using ASP.NET, according to your initiation specifications.

Similarly, you can create custom SharePoint tasks for your workflow. Again, SharePoint Designer automatically generates an ASP.NET form for the task, according to your specifications.

These aspx forms are stored on the SharePoint site with the workflow source files. You can open and customize them as you would any other aspx form.

|  |
| --- |
| **NoteNote:** |
| SharePoint Designer 2007 does not offer integration with InfoPath forms. |

**Deploying Workflows with SharePoint Designer 2007**

Because you author against a specific list, the deployment of workflows created in Office SharePoint Designer is a much simpler process than workflows created in the Visual Studio 2005 Designer for Windows Workflow Foundation. SharePoint Designer handles the deployment of the workflow to the specified list.

SharePoint Designer offers no custom debugging functionality.

Deleting a workflow authored in SharePoint Designer from a list does not delete the actual source files used to compile that workflow in memory. Rather, the workflow is no longer associated with the list, but the source files remain stored in the workflow document library on the site.

In the Windows SharePoint Services object model, workflows authored in SharePoint Designer are indistinguishable from workflows authored in the Visual Studio 2005 Designer for Windows Workflow Foundation.

**Comparing the Visual Studio 2005 Designer for Windows Workflow Foundation and SharePoint Designer 2007**

The following table offers a detailed comparison between the capabilities offered by the Visual Studio 2005 Designer for Windows Workflow Foundation and Office SharePoint Designer 2007, and the workflows you can create with each.

**Table 1. Detailed comparison of capabilities**

|  |  |
| --- | --- |
| **Visual Studio 2005 Designer for Windows Workflow Foundation** | **SharePoint Designer 2007** |
| Can write workflows for Windows SharePoint Services or SharePoint Server | Can write workflows for Windows SharePoint Services or SharePoint Server |
| Code-behind file enables developer to write custom Visual C# or Visual Basic .NET code to express business logic | No code-behind file; workflow rules file declaratively encapsulates business logic |
| Generates workflow markup file | Generates workflow markup file |
| Workflow is authored as a template, which can be associated with multiple sites and lists | Workflow is authored against and data-bound to specific list at design time |
| Workflow markup file, or markup and code-behind files, are compiled into workflow assembly | Workflow markup, workflow rules, and supporting file are stored, uncompiled, in a specific document library on the site |
| Workflow template must be associated with each list on which it is to be available | Association happens when the workflow is authored against the specific list; no later association is necessary or possible |
| Can use any forms technology. For example, ASP forms for Windows SharePoint Services workflows, or InfoPath forms for SharePoint Server workflows | Automatically generates ASP.NET forms, which you can then customize |
| Can include workflow modifications | Workflow modifications are not available |
| Can use custom symmetrical InfoPath forms, which enables Office client integration of custom workflow forms | InfoPath forms integration not available |
| Can author custom activities for inclusion in workflows | Must use activities provided |
| Package workflow assembly and workflow definition as a SharePoint feature, and deploy to the site | Automatically handles deployment to specific list |
| Can use Initiation form to gather information from the user when starting the workflow | Can use Initiation form to gather information from the user when starting the workflow |
| Can use custom forms for users to interact with SharePoint tasks | Can use custom forms for users to interact with SharePoint tasks |
| Visual Studio debugging available | No step-by-step debugging available |
| Can author both sequential and state workflows | Can author only sequential workflows |

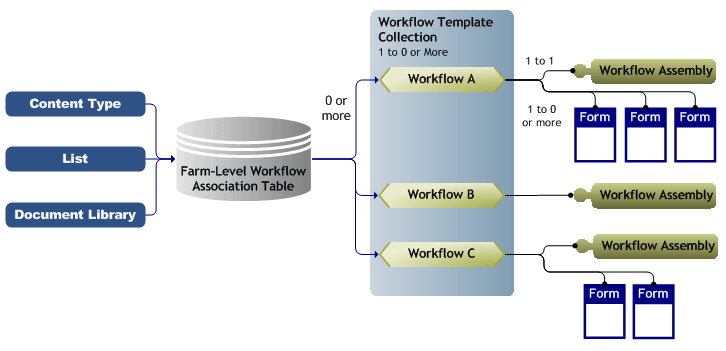
**Workflow Structure in Windows SharePoint Services**

Figure 8 shows how the various elements of the Windows SharePoint Services workflow structure are joined after a workflow template has been created, deployed, and associated with a specific content type, list, or document library.

When you associate a workflow template with a given content type, list, or document library, Windows SharePoint Services writes the association parameter information set by the administrator to a farm-level workflow association table. The entry in this association table links the specific content type, list, or document library to the workflow template definition of the associated workflow. The information in the workflow template definition, in turn, includes references to the workflow assembly itself, as well as any custom workflow forms the workflow uses.

In the case of workflows authored in SharePoint Designer 2007, the Designer automatically writes the association parameter information to the workflow association table, and generates and installs the workflow template definition. Also, the workflow is stored in a site-level workflow document library, not as a compiled assembly, but as the markup and workflow rules source files.

**Figure 8. Workflow component structure in Windows SharePoint Services**



**Using the Workflow Namespace**

After you deploy your workflow solution, you can use the Windows SharePoint Services object model to query workflow processes and programmatically perform workflow actions, such as adding a workflow to a list or starting a workflow for an item.

**Major Microsoft.Windows.SharePoint.Workflow Objects**

The **Microsoft.Windows.SharePoint.Workflow** namespace represents the workflow functionality contained in Windows SharePoint Services.

The **SPWorkflowTemplateCollection** object represents the workflow templates currently deployed on a site. Each **SPWorkflowTemplate** object represents a workflow template and contains properties you can use to get or set information about the template, such as the instantiation data and the history and task lists for the template.

To associate a workflow with a list or document library, use the **SPList.AddWorkflowAssociation** method. This overloaded method takes four parameters:

* Either the workflow template name or ID
* The name you want to give the workflow
* Either the task list name or ID for the task list you want to use with this workflow
* The history list for the workflow

As with adding a workflow through the user interface, this method adds a status column for the workflow to the list. Use the **SPList.RemoveWorkflowAssociation** method to remove a workflow template from a list.

The **SPWorkflowAssociationCollection** object represents the internal workflow associations table for a site; that is, it represents the workflows currently associated on lists across a site collection, along with the association information for each. Each **SPWorkflowAssociation** object represents a workflow template that has been associated with a specific list, and contains properties that return custom information about the association, such as whether the workflow is enabled, whether the workflow has been modified, and the list with which the workflow has been associated, as well as a reference to the **SPWorkflowTemplate** object that serves as a base for this **SPWorkflowAssociation** object.

The **SPWorkflowAssociation.IsDeclarative** property returns **True** for workflows that are stored as uncompiled files, such as those authored in SharePoint Designer.

The **SPWorkflowCollection** represents the workflow instances that have run or are currently running on a given list item. Each **SPWorkflow** object contains properties that return information about the workflow instance, such as whether the workflow has completed, its internal state, and the workflow template on which it is based. In addition, each workflow contains a collection of the tasks for the workflow, **SPWorkflowTaskCollection**.

Use the **SPListItem.Workflows** property to return an **SPWorkflowCollection** object that represents the workflows currently running for that list item.

**Managing Running Workflow Instances Programmatically**

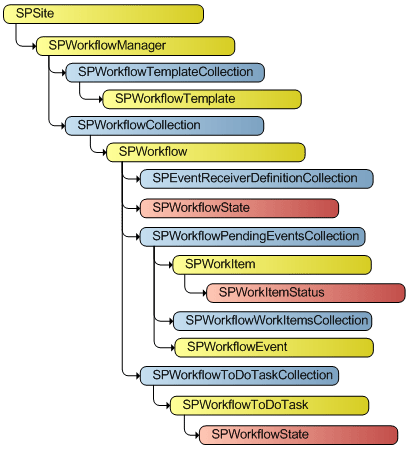
Users interact with the workflows running on items individually, through the Windows SharePoint Services user interface. But Windows SharePoint Services provides functionality for you to centrally control the running instances of workflows across your entire server farm through the object model. Use the **SPWorkflowManager** object to manage the running instances of workflows across the entire farm. The **SPWorkflowManager** object has no equivalent in the user interface. Although the object is accessed through the **SPSite** object, you can use it to administer the workflow instances running on your entire farm. Use the **SPWorkflowManager** object to:

* Start, run, or cancel workflows.
* Return all the workflows currently running on a specific item.
* Cancel all workflow instances based on a specific workflow template within a list.
* Perform other workflow administration operations.

To manually start a specific workflow for an item (that is, a workflow that is not configured to start automatically), use the **SPSite.WorkflowManager.StartWorkflow** method. This method takes three parameters: the name of the list item, the name of the workflow template, and the start event.

Figure 9 shows a hierarchical view of the **SPWorkflowManager** object and the objects it contains.

**Figure 9. SPWorkflowManager object hierarchy**



**Conclusion**

Windows SharePoint Services 3.0 hosts Windows Workflow Foundation, which enables you to attach custom business processes, implemented as sequential or state machine workflows, to SharePoint items. This implementation includes the ability to create custom workflows, complete with forms integration for user interaction. SharePoint Server 2007 extends these workflow capabilities by offering integration with symmetrical InfoPath forms that can be hosted in Office client applications such as Word, PowerPoint, Excel, and InfoPath.

Microsoft provides two powerful tools for authoring workflows for Windows SharePoint Services. The Visual Studio 2005 Designer for Windows Workflow Foundation is an add-in for Visual Studio 2005 that gives you the ability to develop workflow templates, complete with custom code, which you can deploy to multiple sites and list. Alternatively, you can use SharePoint Designer to rapidly develop and deploy list-specific workflows from a predefined list of workflow activities.

In either case, after you deploy your workflow solution, you can use the Windows SharePoint Services object model to query workflow processes and programmatically perform workflow actions